DATA MINER AND ANALYSER FOR E-COMMERCE

A Mini Project Report

submitted by

MOHAMMED AJNAS.K (MES20MCA-2027)

to

the APJ Abdul Kalam Technological University
in partial fulfillment of the requirements for the award of the Degree

of

Master of Computer Applications

Department of Computer Applications

MES College of Engineering
Kuttippuram, Malappuram - 679 582

February 2022

DECLARATION

I undersigned hereby declare that the project report **DATA MINER AND ANALYSER FOR E-COMMERCE**, submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by me under supervision of Mr.Vasudevan T V, Assistant Professor, Department of Computer Applications. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place:Kuttippuram

Date:04-03-22

.....

DEPARTMENT OF COMPUTER APPLICATIONS
MES COLLEGE OF ENGINEERING, KUTTIPPURAM

CERTIFICATE

This is to certify that the report entitled **DATA MINER AND ANALYSER FOR E-COMMERCE** is a bonafide record of the Mini Project work carried out by **MOHAMMED AJNAS.K (MES20MCA-2027)** submitted to the APJ Abdul Kalam Technological University, in partial fulfillment of the requirements for the award of the Master of Computer Applications, under my guidance and supervision. This report in any form has not been submitted to any other University or Institution for any purpose.

Internal Supervisor(s) External Supervisor(s)

Head Of The Department

Acknowledgements

With profound sense of gratitude I wish to express my sincere thanks to Dr K.A.Navas, Principal MES COLLEGE OF ENGINEERING KUTTIPPURAM for giving an opportunity to undertake this project. I have great pleasure in expressing my profound gratitude to Mr. Hyderali K,Assoociate Professor, Head of the Department of Master Of Computer Applications for giving the valuable help during the project work. I gratefully acknowledge my internal guide Mr.Vasudevan T V, Assistant Professor, Department of Master Of Computer Application, MES COLLEGE OF ENGINEERING KUTTIPPURAM for his encouragement till the end of the project. Last but not least, I would like to express my sincere thanks to the God almighty for his constant love and grace that he has bestowed upon us. Finally I thank my parents, family members and beloved friends for their moral support and encouragement without which I have not been able to follow my dreams.

MOHAMMED AJNAS.K (MES20MCA-2027)

Abstract

This system are designed to compare the price of product from Amazon and flipkart, which will help consumers in making decision to choose products that will save their money through online. Considering the customers' busy lifestyle especially those who are living in the city area, most of the consumers prefer to buy their needs through the internet because it saves their time. Besides, consumers always go for the cheaper price in purchasing products therefore by using this system, customers do not want to search from one site to another sites for the same product. They can just check it from the price comparison website itself and decide where they should buy the products they need. This project, named as DATA MINER AND ANALYSER FOR E-COMMERCE is the place where shoppers could find the great deals on the products. To obtain best deals from Price comparison websites web scrapping techniques are used to fetch detailed information. This way, project aims to provide solution for online customers to buy products at good deal and save their valuable time, effort, and money.

Keywords:web scrapping, e-commerce, Flask, Price comparison

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Every customer looks for the best deals discounts before buying any product. Nowadays before purchasing anything the buyers do some online research of the products on the internet. One of the major factors which lead to purchasing of any product is cost or pricing. The buyers tend to compare prices before purchasing any product. But since it is very difficult to visit each every website for price comparison, there needs to be a solution to automate this process. This project proposed here gathers information on product prices and details from various websites presents it to the users.The data is scraped from flipkart and Amazon.

The backend system consists of techniques web scrapping.Web scrapping is a technique that is used to extract information in the human readable format and display it on destination terminal.Web scrapping essentially consists of two tasks: first is to load the desired web page and second is to parse HTML information of the page to locate intended information. In this system Scrapping is done using python as it provides rich set of libraries to address these tasks. “requests” is used to load the URLs and “Beautiful soup” library is used to parse the web page. After scrapping the products information from different e-commerce websites, the data is displayed on the website. The frond end consists of Main website. The client searches for the required product in search bar and query is fired in local database i.e., sqlite3. The website is designed using Flask web framework which is written in python. Required results are retrieved and displayed on Main website from database SQLite3. The client can then compare prices of products that are available on e-commerce websites. A soon as client

selects on best deal according to him, he will be redirected to the original ecommerce website

The users can then choose to buy from the best options available. Even Ecommerce traders can use this price comparison website to study their competitors and form new strategies accordingly to attract new customers stay ahead of their competitors

1.1.1 Motivation

In the current era of online business, ecommerce has become a huge market for the people to buy goods online. Increasing use of smart devices and other mediums has paved the way for users to buy products almost from anywhere. This has increased involvement of online buyers evolving e-commerce business. These large numbers of ecommerce websites put users in turmoil to search and choose to buy a single product from multiple ecommerce websites.So we need a solution to helps online users to grab best deal for their product from multiple ecommerce websites on single web interface. This will in turn save users time, money, and efforts to find the same product prices on different ecommerce websites.This is the prime motivation behind making a price comparison website

1.2 Objective

Our objective here is to build a system which will provide the customer with the same product from different websites at one place. It will help the customer compare and decide as to which will be the best option for them thus making their shopping experience faster and more efficient. The DATA MINER AND ANALYSER FOR E-COMMERCE SYSTEM will recommend the appropriate product to the customer in an automated manner which will save time and money

1.3 Contribution

Comparison of product prices from different e-commerce websites and result is displayed on single web interface. This website aims at providing the best possible deal to the users for the required product by comparing the product price and displaying the minimum price from various E-commerce websites such as Amazon and Flipkart

1.4 Report Organization

The project report is divided into four sections. Section 2 describes literature survey. Section 3 describes the methodology used for implementing the project. Finally Section 4 gives the conclusion.

Chapter 2

Literature Survey

Web Scraping with Python Written By Richard Lawson. This book is aimed at developers who want to build reliable solutions to scrape data from websites. It is assumed that the reader has prior programming experience with Python. Anyone with general knowledge of programming languages should be able to pick up the book and understand the principles involved

Written By S. Rajendar, K. Manikanta, M. Mahendar, Assistant Prof. (Mrs.) K. Madhavi,Department of Computer Science and Engineering, St. Peter's Engineering College, Hyderabad.This paper discuss about Comparison of product prices from different ecommerce websites and technologies are used.

Chapter 3

Methodology

3.1 Introduction

DATA MINER AND ANALYSER FOR E-COMMERCE is implemented Using Flask and get the data from amazon and flipkart by web scrapping.The data stored in database SQLite.

3.1.1 Web Scrapper

Web Scrapping is used to extract HTML data from URL's and use it for personal purpose. As this is price comparison website, data is scrapped from multiple e-commerce websites. In this system, Scrapping is done using python libraries like requests and beautifulsoup4. Beautifulsoup4 is a python library which is used for parsing html pages. Using these, product information from different e-commerce sites is scrapped and stored in database.

3.1.2 Pycharm

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. Python is a broadly useful deciphered, intelligent, object-situated, and significant level programming language. It gives special importance to code legibility and makes the computer specialist tasks easy by writing code in a small number of lines

3.1.3 Flask And SQLite3

Flask is a web framework that provides libraries to build lightweight web applications in python.It is free and open source.SQLite3 is database used to store the data from web scrapping.

3.2 Agile Methodology

The Agile methodology is a way to manage a project by breaking it up into several phases. It involves constant collaboration with stakeholders and continuous improvement at every stage. Once the work begins, teams cycle through a process of planning, executing, and evaluating. Continuous collaboration is vital, both with team members and project stakeholders.

3.2.1 User Story

UserStoryID	As a <type of user>	I want to	So that I can
1	User	login	login successful with correct username and password
2	User	Registration	user's can register with this webpage
3	User	Search Product	Search product from ecommerce website and add to the database
4	User	View the product detail	We can see the product detail from different ecommerce website
5	User	Go to the product page	We can go to the product page which you like and you can buy

Figure 3.1: User Story

3.2.2 Product Backlog

User Story ID	Priority <High/Medium/Low>	Size (Hours)	Sprint <#>	Status <Planned/In progress/Completed>	Release Date	Release Goal
1	Medium	2	1	Completed	28/12/2021	Table design
2	High	3		Completed	31/12/2021	Form design
3	High	5		Completed	08/01/2022	Basic coding
4	High	9	2	Completed	16/01/2022	Collecting data from flipkart and amazon
5	Medium	6		Completed	21/01/2022	Add the data to database
6	Medium	5	3	Completed	03/02/2022	Taking accurate data from database
7	Medium	5	4	Completed	12/02/2022	Testing data
8	High	5		Completed	17/02/2022	Output generation

Figure 3.2: Product Backlog

3.2.3 Project Plan

User Story ID	Task Name	Start Date	End Date	Days	Status
1	Sprint 1	26/12/2021	28/12/2021	2	Completed
2		29/12/2021	31/12/2021	3	Completed
3		03/12/2021	08/01/2022	5	Completed
4	Sprint 2	09/01/2022	16/01/2022	8	Completed
5		17/01/2022	21/01/2022	5	Completed
6	Sprint 2	23/01/2022	03/02/2022	12	Completed
7	Sprint 4	04/02/2022	12/02/2022	9	Completed
8		13/02/2022	17/02/2022	5	Completed

Figure 3.3: Project Plan

3.2.4 Sprint Backlog Plan

Backlog Item	Status & completion date	Original estimate in hours	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10	Day11	Day12	Day13	Day14
User story #1,#2,#3		hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs
Table design	28/12/2021	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Form design	31/12/2021	3	0	0	0	1	1	1	0	0	0	0	0	0	0	0
Coding	08/01/2021	5	0	0	0	0	0	0	0	0	0	1	1	1	1	1
User story 4,5																
Collecting data from flipkart and amazon	16/01/2022	9	2	2	1	1	0	1	0	2	0	0	0	0	0	0
Add the data to database	21/01/2022	6	0	0	0	0	0	0	0	0	2	1	1	1	1	0
User story 6																
Collecting data from amazon	27/01/2022	5	2	1	1	0	1	0	0	0	0	0	0	0	0	0
User story 7,8																
Testing data	12/02/2022	5	0	0	0	0	0	2	0	1	2	0	0	0	0	0
Output generation	17/02/2022	5	0	0	0	0	0	0	0	0	0	2	0	1	1	1
Total		40	5	4	2	2	2	4	0	3	4	4	2	3	3	2

Figure 3.4: Sprint Backlog Plan

3.2.5 Sprint Backlog Actual

Backlog Item	Status & completion date	Original estimate in hours	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10	Day11	Day12	Day13	Day14
User story #1,#2,#3,#4		hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs
Table design	28/12/2021	3	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Form design	31/12/2021	2	0	0	0	1	1	1	0	0	0	0	0	0	0	0
Coding	08/01/2021	5	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Total		10	1	1	0	1	1	1	0	0	0	1	1	1	1	1

Figure 3.5: Sprint1 Actual

Backlog Item	Status & completion date	Original estimate in hours	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10	Day11	Day12	Day13	Day14
User story #4,#5		hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs
Collecting data from flipkart and amazon	16/01/2022	9	2	2	1	1	0	1	0	2	0	0	0	0	0	0
Add the data to database	21/01/2022	6	0	0	0	0	0	0	0	0	2	1	1	1	1	0
Total		15	2	2	1	1	0	1	0	2	2	1	1	1	1	0

Figure 3.6: Sprint2 Actual

Backlog Item	Status & completion date	Original estimate in hours	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10	Day11	Day12	Day13	Day14
User story #6		hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs
Taking accurate data from database	27/01/2022	5	2	1	1	0	1	0	0	0	0	0	0	0	0	0
Total		5	2	1	1	0	1	0	0	0	0	0	0	0	0	0

Figure 3.7: Sprint3 Actual

Backlog Item	Status & completion date	Original estimate in hours	Day1	Day2	Day3	Day4	Day5	Day6	Day7	Day8	Day9	Day10	Day11	Day12	Day13	Day14
User story #7,#8		hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	hrs	Hrs
Testing data	12/02/2022	5	0	0	0	0	0	2	0	1	2	0	0	0	0	0
Output generation	17/02/2022	5	0	0	0	0	0	0	0	0	0	2	0	1	1	1
Total		10	0	0	0	0	0	2	0	1	2	2	0	1	1	1

Figure 3.8: Sprint4 Actual

Chapter 4

Conclusions

The website provides users with useful information that will help them making informed decision. With this DATA MINER AND ANALYSER FOR E-COMMERCE, it solves the problems of the working people to check on the price before buying products. This website will facilitate users to analyze prices that are present on different e-commerce shopping websites like Amazon and Flipkart so that they get to know the cheapest price of product with best deal. This will surely save buyers efforts and valuable time. Ultimately, this will bring together strategies, best offers and deals from all leading online stores and will help buyers to shop online.

References

[1] **Amazon**: https://www.amazon.in/.

[2] **Flipkart**: https://www.flipkart.com/.

[3] **beautifulsoup4**: https://pypi.org/project/beautifulsoup4/.

[4] **J**os´e Ignacio Fern´andez-Villamor, Jacobo Blasco-Garc´ıa, Carlos ´A.Iglesias, Mercedes Garijo "A Semantic Scrapping Model for Web Resources" Spain.

[5] **Benoit Marchal, Tip: Convert from HTML to XML with HTML Tidy** (18 Sep 2003) : http://www-106.ibm.com/developerworks/library/x-tiptidy.html?ca=dgr-lnxw02TidyUp

Appendix

Source Code

```
import requests
import urllib
from flask import Flask, render_template, request, flash, redirect, url_for, session
import sqlite3
from bs4 import BeautifulSoup
import phone ,shoes,lap,watch,tab
import os
import sys
import random


import csv
app = Flask(__name__)
app.secret_key="123"

def get_proxies():
   proxy_url="https://github.com/clarketm/proxy-list/blob/master/proxy-list-raw.txt"
   r = requests.get(proxy_url)
   soup = BeautifulSoup(r.content, "html.parser").find_all("td",{"class":"blob-code blob-code-inner js-file-line"})
   proies = [proxy.text for proxy in soup]
   return proies
def get_random_proxy(proxies):
   return {"https//":random.choice((proxies))}

proxies=get_proxies()
working = []
def get_working_proxies():

   for i in range(30):
      proxy = get_random_proxy(proxies)
      print(f"using {proxy}...")
      try:
         r=requests.get("http://www.google.com/",proxies=proxy, timeout=3)
         #r=
               requests.get("https://www.amazon.in/s?k=iphone+13&crid=CU5LJWR9SK3S&sprefix=ipho%2Caps%2C1541&ref=nb_sb_ss_ts-doa-p_1_4",proxies
               timeout=3)
         if r.status_code == 200:
            print('200')
            working.append(proxy)
            break
      except:
         pass
   return working
```

Appendix

```
proxy=get_working_proxies()
print(proxy[0])
proxy =proxy[0]
print(proxy)
print(working)

user_agent_list = [
'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:77.0) Gecko/20100101 Firefox/77.0',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/83.0.4103.97 Safari/537.36',


   'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Silk/44.1.54 like Chrome/44.0.2403.63
        Safari/537.36'
]

#for i in range(1,4):
#Pick a random user agent
user_agent = random.choice(user_agent_list)
#Set the headers
headers = {'User-Agent': user_agent}
print("hi")
s=['home1.html']
def amazon(pro,cho):
   if cho == 'phone':
      print("amazon")
      print(pro)
      print(cho)
      st =300
      if st == 200:
         return st
      else:

         return st

   else:
      print("amazon")

def flip(pro,cho):
   if cho == 'phone':
      print("flip")
      print(pro)
      print(cho)
      sc=200
      return sc


   else:
      print("flip")
@app.route("/")
def index():

   if 'name' in session:
      return render_template("userpage.html")
   else:
      return render_template("home1.html")
@app.route("/loginpage")
def loginpage():
   return render_template("home.html")


@app.route('/login',methods=["GET","POST"])
def login():
   if request.method=='POST':
```

Appendix

```
        email=request.form.get('email')
        password=request.form.get('password')
        con=sqlite3.connect("db.db")
        con.row_factory=sqlite3.Row
        cur=con.cursor()
        cur.execute("select * from user where email=? and password=?",(email,password))
        data=cur.fetchone()



        if data:
            session.clear()
            session["email"]=data["email"]
            session["password"]=data["password"]
            s1='userpage.html'
            s.clear()
            s.append(s1)
            return render_template("userpage.html")
        else:
            flash("Username and Password Mismatch","danger")
    return redirect(url_for("home"))

@app.route('/register',methods=['GET','POST'])
def register():
    print(request.method)
    if request.method=='POST':
        print("hi")

        try:
            name=request.form.get('name')
            email=request.form.get('email1')
            password=request.form.get('password1')
            print(name)
            con=sqlite3.connect("db.db")
            cur=con.cursor()
            cur.execute("insert into user(name,email,password)values(?,?,?)",(name,email,password))
            con.commit()
            flash("Record Added Successfully","success")
        except:
            flash("Error in Insert Operation","danger")
        finally:
            return render_template("home.html")
            con.close()

    return render_template('register.html')
@app.route('/logout')
def logout():
    session.clear()
    s1='home1.html'
    s.clear()
    s.append(s1)
    return render_template("home.html")
@app.route('/pro_detail')
def pro_detail():
    con = sqlite3.connect("db.db")
    cur = con.cursor()
    cur.execute("select * from pro_de")
    s = cur.fetchall()
    path = 'C:/Users/DELL/PycharmProjects/mini_project/static/images'
    os.chdir(path)
    print("lkjjlnbk")
    count = 1
    for img in s:
```

Appendix

```
        with open("{}.jpg".format(count), "wb") as f:
            f.write(img[8])
            count = count + 1
    path = 'C:/Users/DELL/PycharmProjects/mini_project'
    os.chdir(path)
    return render_template('pro_detail.html', val=s)


def laptop():
    con = sqlite3.connect("db.db")
    cur = con.cursor()
    cur.execute("select * from ")
    s = cur.fetchall()
    return render_template('pro_lap.html', val=s)

@app.route('/view')
def view():

    id = request.args.get('id')
    cwd = os.getcwd()
    print(cwd)
    s=id
    print(s)

    con = sqlite3.connect('db.db')
    cur = con.cursor()
    cur.execute("select * from pro_de where pid=?",(s,))
    s = cur.fetchall()
    return render_template('view.html',val=s)

@app.route('/view1')
def view1():

    id = request.args.get('id')
    cwd = os.getcwd()
    print(cwd)
    s=id
    print(s)

    con = sqlite3.connect('db.db')
    cur = con.cursor()
    cur.execute("select * from lap where pid=?",(s,))
    s = cur.fetchall()
    return render_template('view1.html',val=s)
view_v=[]
@app.route('/a_price')
def a_price():
    print("hi")
    print(view_v)
    id = request.args.get('id1')
    view_v.append(id)
    print(id)
    s=id
    con = sqlite3.connect("db.db")
    cur = con.cursor()
    cur.execute("select * from a_phone where pid=?", (s,))
    print("ooooo")
    s = cur.fetchall()
    return render_template('price1.html',val=s)
@app.route('/f_price')
def f_price():
    print("hi")
    print(view_v)
```

Appendix

```
    id = request.args.get('id1')
    view_v.append(id)
    print(id)
    s=id
    con = sqlite3.connect("db.db")
    cur = con.cursor()
    cur.execute("select * from f_phone where pid=?", (s,))
    print("ooooo")
    s = cur.fetchall()
    return render_template('price2.html',val=s)
@app.route('/price1',methods=['POST','GET'])
def price1():
    id = request.args.get('id')
    print(id)
    epr=request.form['mail1']
    if 'name'in session:
        print(session['name'])
        amazon='amazon'
        con = sqlite3.connect('db.db')
        cursor = con.cursor()
        cursor.execute(
            "create table if not exists price(pid integer primary key,email text,price text,link text,site text)")
        cursor.execute("insert into price(email,price,link,site)values(?,?,?,?)", (session['name'], epr,id,amazon))
        con.commit()
        con.close()
        return redirect(url_for('a_price',id=id))
    else:
        print("no")
        return redirect(url_for('loginpage'))
@app.route('/price2',methods=['POST','GET'])
def price2():
    id = request.args.get('id')
    print(id)
    epr=request.form['mail2']
    if 'name'in session:
        print(session['name'])
        flip1='flipkart'
        con = sqlite3.connect('db.db')
        cursor = con.cursor()
        cursor.execute(
            "create table if not exists price(pid integer primary key,email text,price text,link text,site text)")
        cursor.execute("insert into price(email,price,link,site)values(?,?,?,?)", (session['name'], epr,id,flip1))
        con.commit()
        con.close()
        return redirect(url_for('f_price',id=id))
    else:
        print("no")
        return redirect(url_for('loginpage'))

@app.route('/aphone')
def aphone():
    con = sqlite3.connect("db.db")
    cur = con.cursor()
    cur.execute("select * from a_phone")
    s = cur.fetchall()
    cur.execute("select * from f_phone")
    m = cur.fetchall()
    return s,m
@app.route('/fphone')
def fphone():
    con = sqlite3.connect("db.db")
    cur = con.cursor()
    cur.execute("select * from a_phone")
```

```
    s = cur.fetchall()
    cur.execute("select * from f_phone")
    m = cur.fetchall()
    return s,m
@app.route('/', methods=['POST'])
def getvalue():
    print(s[0])
    product = request.form['name_in']
    choice = request.form['choice']
    if choice == 'phone':
        n='1'
        st =phone.a_phone(product,choice,headers,proxy)
        if st==200:
            sc=phone.f_phone(headers,proxy)
            if sc==200:
                print("popup")
                flash("Server ok", "danger")
                sm,sk=aphone()
                return render_template('aphone.html',val1=sm,val2=sk)
            else:
                flash("Server busy", "danger")
                return render_template(s[0])
        else:
            flash("Server busy", "danger")
            return render_template(s[0])
    elif choice=='laptop':
        st = lap.f_lap(product, choice,headers,proxy)
        if st == 200:
            sc = lap.a_lap(headers,proxy)
            if sc == 200:
                print("popup")
                flash("Server ok", "danger")
                sm, sk = fphone()
                return render_template('alap.html', val1=sm, val2=sk)
            else:
                flash("Server busy", "danger")
                return render_template(s[0])
        else:
            flash("Server busy", "danger")
            return render_template(s[0])
    elif choice=='watches':
        st = watch.f_watch(product, choice,headers,proxy)
        if st == 200:
            print("popup")
            flash("Server ok", "danger")
            sm, sk = aphone()
            return render_template('aphone.html', val1=sm, val2=sk)
        else:
            flash("Server busy", "danger")
            return render_template(s[0])


    elif choice=='shoes':
        n=1
        st = shoes.f_shoes(product, choice,headers,proxy)

        if st == 200:
            print("popup")
            flash("Server ok", "danger")
            sm,sk = aphone()
            return render_template('aphone.html',val1=sm,val2=sk)
        else:
            flash("Server busy", "danger")
```

Appendix

```
        return render_template(s[0])

   else:
      st = tab.f_tab(product, choice,headers,proxy)
      if st == 200:
         print("popup")
         flash("Server ok", "danger")
         sm, sk = aphone()
         return render_template('aphone.html', val1=sm, val2=sk)
      else:
         flash("Server busy", "danger")
         return render_template(s[0])

   #flip(product,choice)



app.run()
```

Database Design

Attribute Name	Datatype	Width	Description
Pid	Text		Primary Key
Name	Text		
Email	Text		
Password	Text		

Table A.1: user

Attribute Name	Datatype	Width	Description
pid	Integer		Primary Key
name	Text		
price	Text		
link	Text		

Table A.2: aphone

Attribute Name	Datatype	Width	Description
pid	Integer		Primary Key
name	Text		
price	Text		
link	Text		

Table A.3: fphone

Attribute Name	Datatype	Width	Description
pid	Integer		Primary Key
name	Text		
flipkart price	Text		
link	Text		
amazon price	Text		
a link	Text		

Table A.4: fphone

Attribute Name	Datatype	Width	Description
pid	Integer		Primary Key
name	Text		
colour	Text		
rom	Text		
ram	Text		
rating	Text		
price	Text		
link	Text		

Table A.5: phone

Attribute Name	Datatype	Width	Description
pid	Integer		Primary Key
name	Text		
a price	Text		
a rating	Text		
a link	Text		
f price	Text		
f rating	Text		
f link	Text		
photo	Text		

Table A.6: productdetail

Dataflow Diagram

Figure A.1: LEVEL 0

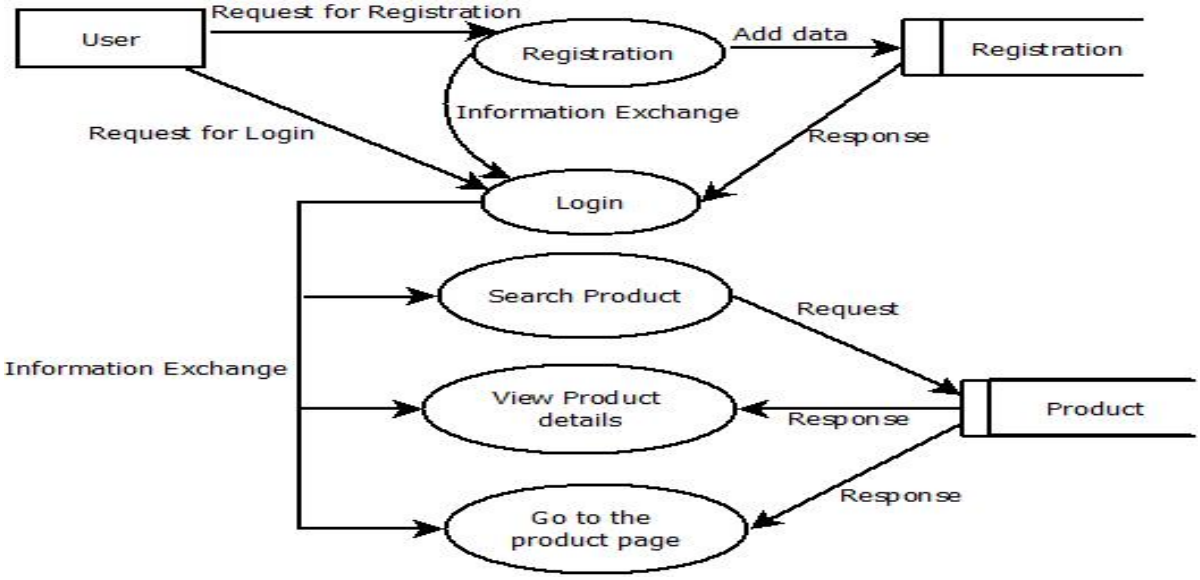

Figure A.2: LEVEL 1.1

User Interface

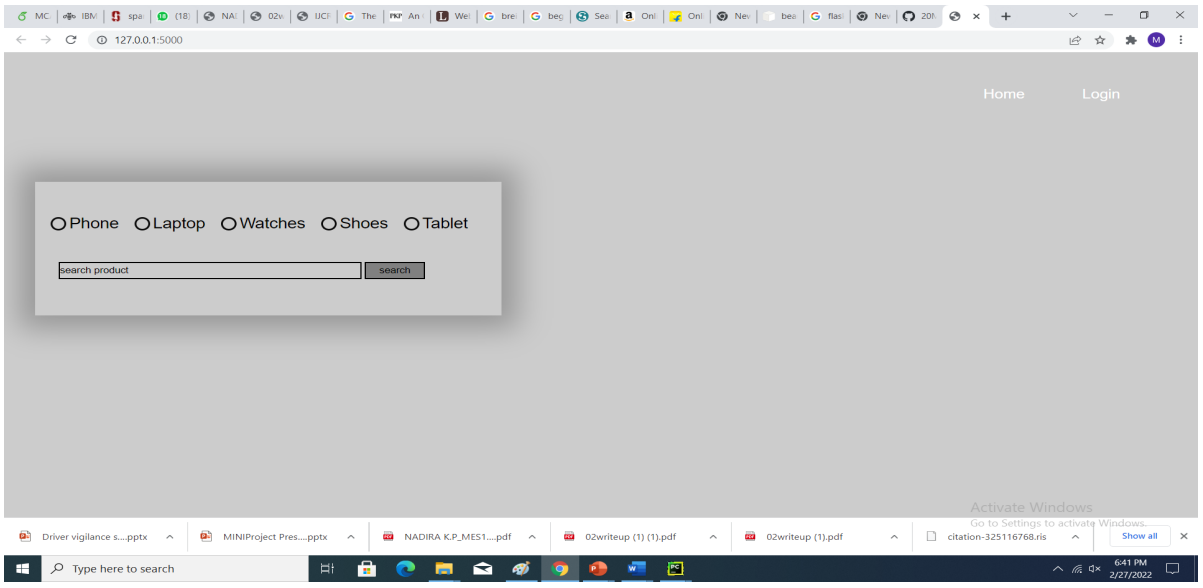

Figure A.3: User Interface 1

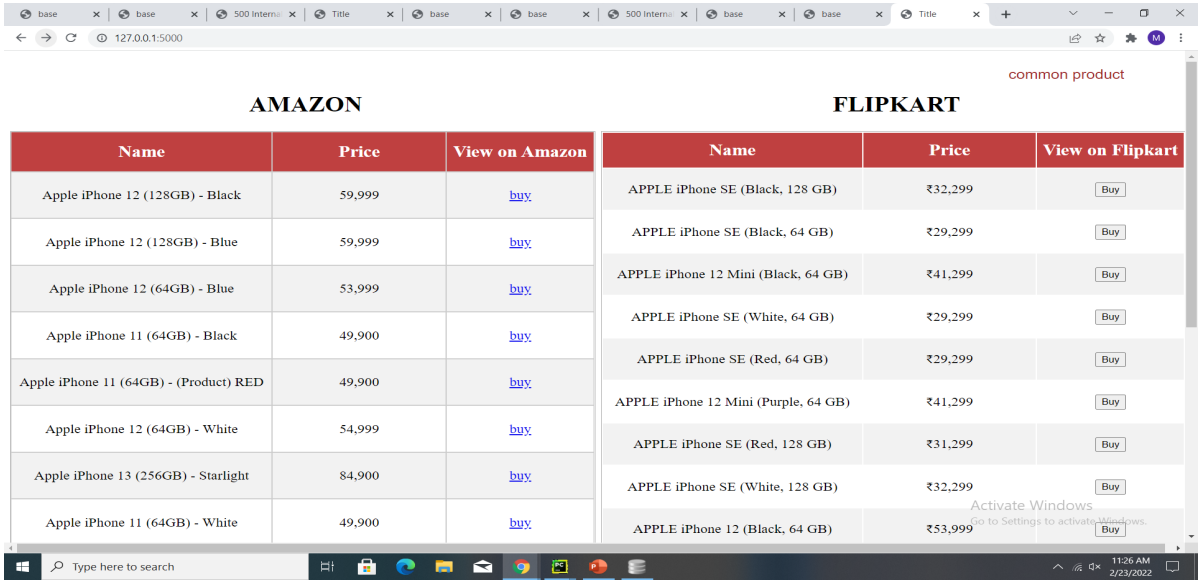

Figure A.4: User Interface 2

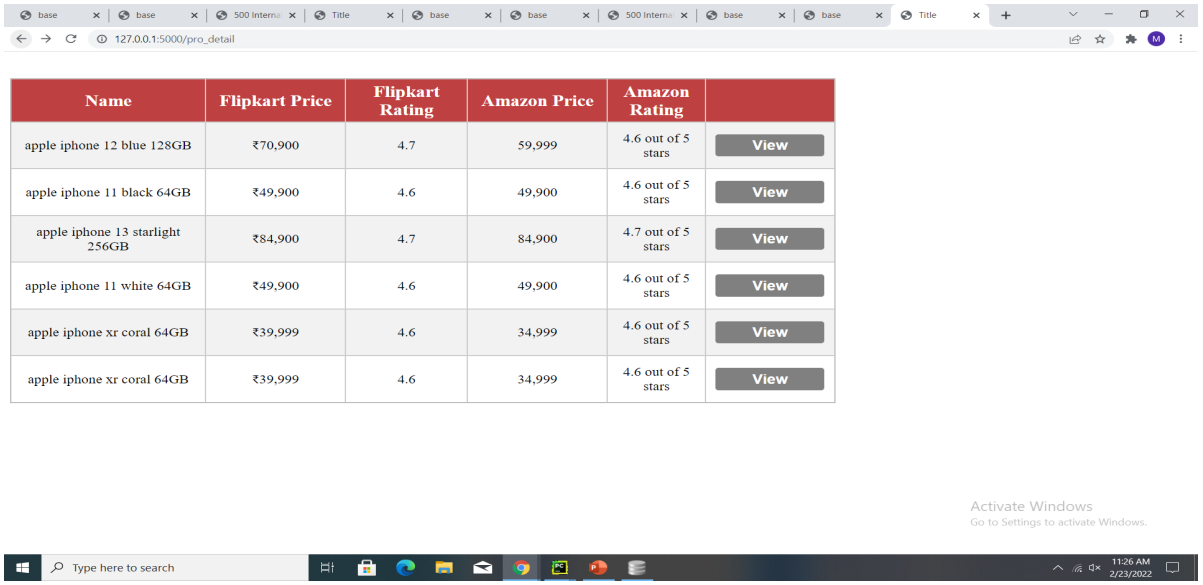

Figure A.5: User Interface 3

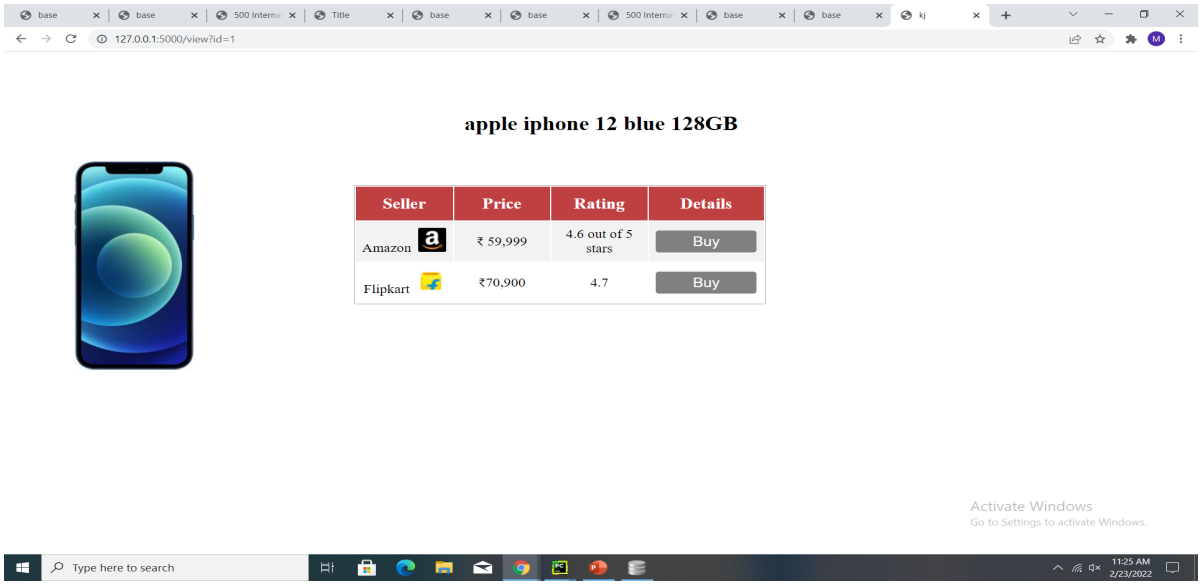

Figure A.6: User Interface 4